Available online at www.sciencedirect.com

ScienceDirect

SoftwareX 3–4 (2015) 27–31

SoftwareX

www.elsevier.com/locate/softx

Grace: A cross-platform micromagnetic simulator on graphics processing units

Ru Zhu

Physics Department, Graceland University, Lamoni, IA 50140, USA

Received 4 April 2015; received in revised form 2 November 2015; accepted 3 November 2015

Abstract

A micromagnetic simulator running on graphics processing units (GPUs) is presented. Different from GPU implementations of other research groups which are predominantly running on NVidia's CUDA platform, this simulator is developed with C++ Accelerated Massive Parallelism (C++ AMP) and is hardware platform independent. It runs on GPUs from venders including NVidia, AMD and Intel, and achieves significant performance boost as compared to previous central processing unit (CPU) simulators, up to two orders of magnitude. The simulator paved the way for running large size micromagnetic simulations on both high-end workstations with dedicated graphics cards and low-end personal computers with integrated graphics cards, and is freely available to download.

Keywords: Micromagnetism; Simulation; GPU; C++ AMP

Code metadata

Current code version	*Version 0.01*
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-15-00006
Legal Code License	*GNU Public License*
Code versioning system used	*git*
Software code languages, tools, and services used	*C++ with C++ AMP acceleration*
Compilation requirements, operating environments & dependencies	*Microsoft Visual Studio 2012 or later; Microsoft Windows 7 or later; C++ AMP FFT Library 1.0*
If available Link to developer documentation/manual	https://github.com/cygnusc/grace/blob/master/Grace.0.01/README.txt
Support email for questions	*zhu@ graceland.edu*

1. Motivation and significance

Micromagnetic simulators are critical tools to study magnetic dynamics and develop new magnetic devices. Central Processing Unit (CPU) based simulators such as OOMMF [1] and magpar [2] are widely used in academic research and industrial applications. The most time-consuming part of micromagnetic simulation is the evaluation of the demagnetization field. A brute force evaluation of a micromagnetic sample with N computational cells results in a time complexity of $O(N^2)$. Thanks to the application of fast numeric methods, the time complexity may be reduced to O(NlogN) using fast Fourier transforms (FFT) [3] or O(N) using the fast multipole method (FMM) [4]. Still, the simulation can be slow in cases with large input problem sizes, as the processing power of a CPU is limited.

Recently, several research groups have reported implementations of micromagnetic simulators on graphics processing units (GPUs) [5–9]. The purpose is to utilize the high computing power of the GPU to speed up the simulation. At the same time, the cost of a GPU (usually less than $1000 for a high-end product) is much less than that of CPU-based clusters. Furthermore,

E-mail address: zhu@graceland.edu.

http://dx.doi.org/10.1016/j.softx.2015.11.001

the FFT algorithm that is used to evaluate the demagnetization field can be easily adopted on GPUs. In fact they are usually included in numeric libraries developed by hardware vendors.

The previously mentioned GPU simulators are all based on NVidia's Compute Unified Device Architecture (CUDA) which limits their applications to NVidia GPUs. Simulators written in CUDA cannot run on GPUs manufactured by other vendors, such as AMD or Intel. Given that these other GPUs (AMD Radeon, FirePro, Intel Iris and Xeon Phi) are popular on professional workstations and personal computing devices, it is desirable to develop a micromagnetic simulator that is not only GPU-accelerated but also hardware cross-platform.

In this paper, Grace, a cross-platform micromagnetic simulator is described which has a speed-up factor of up to two orders of magnitude with respect to CPU calculation for large problems sizes. Section 2 discusses the formulation of micromagnetic simulation. Section 3 describes the implementation of the formulation on GPU. In Section 4 the potential application of the simulator is discussed; the μMag standard problem #3 and #4 [10,11] are used to validate the calculation result. The performance of this simulator is also evaluated. In the end, Section 5 summarizes the paper and discusses potential future work.

2. Software description

2.1. Software architecture

In magnetic nanostructure, the dynamics of magnetization are affected by the effective magnetic field H_{eff} *calculated from the magnetic energy density:*

$$\vec{H}_{eff} = -\frac{\delta \varepsilon}{\delta \vec{M}} = \vec{H}_{exch} + \vec{H}_{anis} + \vec{H}_{demag} + \vec{H}_{extern}$$

where $\frac{\delta \varepsilon}{\delta \vec{M}}$ is the functional derivative of ε with respect to $\vec{M}$, $\vec{H}_{exch}$ is the exchange field and $\vec{H}_{anis}$ is the anisotropy field. A detailed version of how to calculate each term in the effective field can be found in [12].

The Landau–Lifshitz–Gilbert (LLG) equation that governs the magnetic dynamics in the low damping limit is [13]

$$\frac{d\vec{M}}{dt} = -\frac{\gamma}{1+\alpha^2}(\vec{M} \times \mu_0 H_{eff}) - \frac{\alpha\gamma}{(1+\alpha^2)M_s}[\vec{M} \times (\vec{M} \times \mu_0 \vec{H}_{eff})]$$

where α is the damping constant, and γ is the gyromagnetic ratio.

The most critical step in micromagnetic simulation, as mentioned before, is the calculation of the demagnetization field. The direction calculation for N sources at N observers requires a computing time of $O(N^2)$. But since the demagnetization field is actually the convolution of magnetizations and demagnetization tensors in a regular discretization of material, the computation time can be reduced to O(N logN) by applying the discrete convolution theorem and FFT. Non-periodic boundary conditions can be used by adapting it to the zero-padding method [3]. On the other hand, the exchange field calculation is done with a six-neighbor scheme [14]. The time integration of the LLG equation is implemented with the Euler method.

The backbone hardware that accelerates the simulation is a GPU. As opposed to a CPU, which has a limited number of Arithmetic Logic Units (ALUs) with a complicated control unit, the GPU has a much greater number of ALUs but less complicated control for each ALU. As a result, a GPU is suitable for computing-intensive, highly parallelized, and simple algorithms. That is why large scale micromagnetic simulations, with the aid of the FFT algorithm, are ideal cases in which GPU acceleration can be applied.

The software platform is C++ Accelerated Massive Parallelism (C++ AMP) [15], which is an open specification library developed by Microsoft for implementing data parallelism directly from C++. The FFT algorithm used to calculate the demagnetization field is based on the C++ AMP FFT library [16]. At large input sizes, this library can deliver two orders of magnitude performance gain compared to a CPU-based FFT library such as FFTW. This ensures the optimized performance of the simulator.

2.2. Software functionalities

The simulator features a simple and straight-forward input file, two output files and a gnuplot script file to visualize the simulation results, as shown in Fig. 1. Both the input and output files are in plain text format to store data for the best compatibility.

In the script file, a user can specify the geometry of the system simulated, as well as the size of timesteps and the total number of simulated timesteps. The default discretization is 1 nm × 1 nm × 1 nm but that can be modified. Two output files are written, one recording the average magnetization of the sample and the other one recording the magnetizations of evenly sampled computational cells. The interval of writing simulation results data to output files can be specified, e.g. one write operation is executed every 10 timesteps. After the data are saved, the included gnuplot script can be run to visualize the time evolution of magnetizations. Userscan also drag the visualization panel to view the magnetizations from any angle.

3. Illustrative examples

μMag standard problem #3 [10] was used to validate the simulation result. The cubic sample was discretized to 10 × 10 × 10 cells, and the transition point from the flower state to the vortex state was found to be near $l = 8.47 l_{ex}$, where l is the edge length of the cube and $l_{ex} = \sqrt{\frac{2A}{\mu_0 M_s^2}}$ is the intrinsic length scale. This result agreed with reports from [9]. The magnetizations of the cubic particle after the relaxation were shown in Figs. 2 and 3.

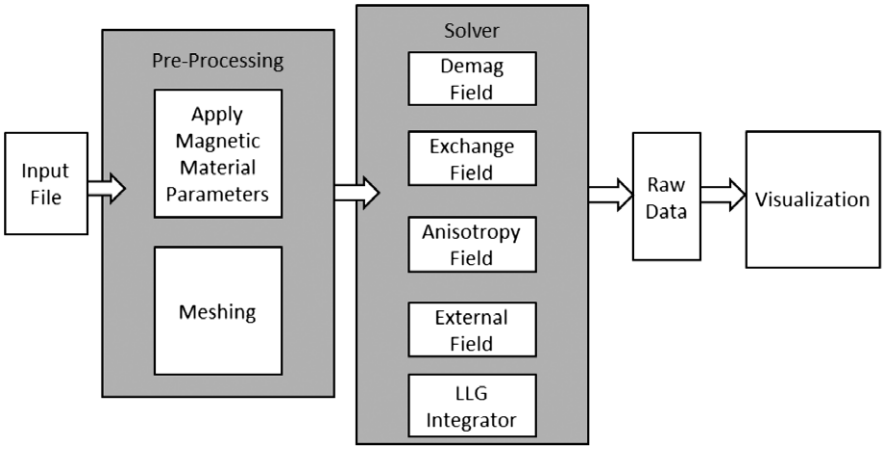

Fig. 1. Overview of the architecture of Grace.

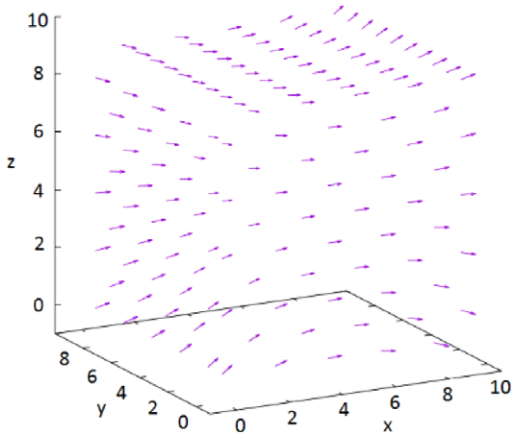

Fig. 2. In μMag standard problem #3, the magnetization of cubic particles in the flower state at $l = 8.47 l_{ex}$.

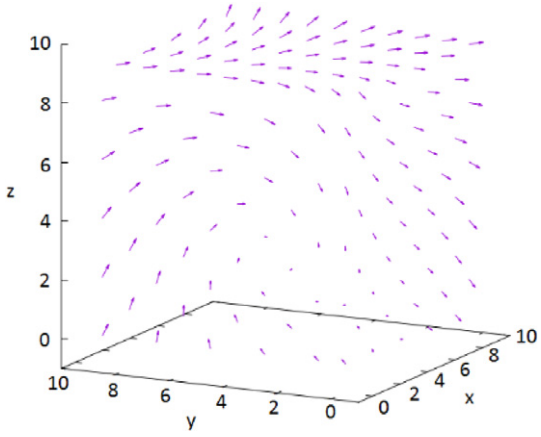

Fig. 3. In μMag standard problem #3, the magnetization of cubic particles in the flower state at $l = 8.6 l_{ex}$.

The μMag standard problem #4 [11] was also used to illustrate the usage of this software and to validate the calculation result. In this problem a thin film sample is divided

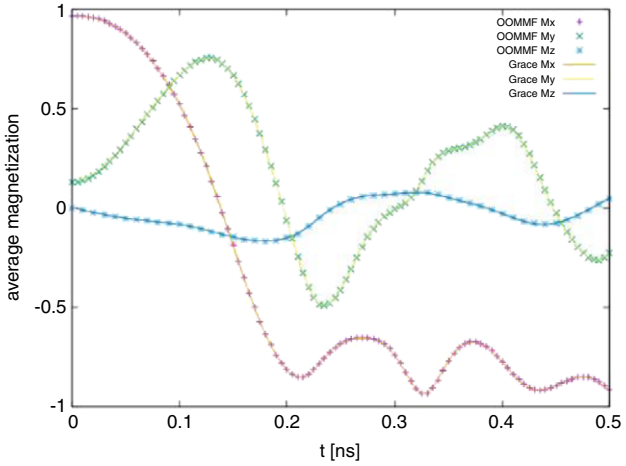

Fig. 4. Average magnetization versus time during the reversal in μMag standard problem #4, field 1. OOMMF simulation results are also presented for comparison.

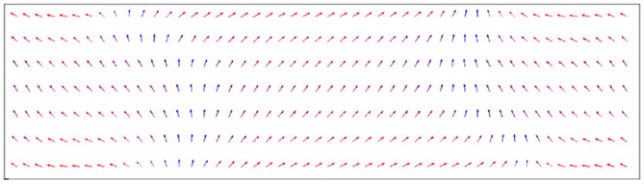

Fig. 5. Magnetization distribution when M_X first crosses zero in μMag standard problem #4, field 1. The domain wall can be clearly seen in the left 1/3 and 2/3 of the sample.

into $500 \times 125 \times 3$ cells, each cell with a size of 1 nm × 1 nm × 1 nm. The sample has an exchange constant of $A = 1.3 \times 10^{-11}$ J/m, a saturation magnetization of $M_S = 8.0 \times 10^5$ A/m, and no anisotropy. Before applying external fields to reverse the magnetization, the system is relaxed to the S-state by setting a large damping constant. Then the test was carried out using a field 1 of (−24.6 mT, 4.3 mT, 0 mT). The damping constant α is set to 0.02 for both tests. The input file is as follows:

```
-simulation 10000              1000000  5e-7
#          writeInterval     timesteps dt
-rectang 1000  250  6
#         nx    ny   nz
-material 0.02  1.3e-11   800      0        0        0   0   0
#          alpha A (J/m) M_init.x M_init.y M_init.z Hkx Hky Hkz
-readInitState
-externfield -19.576 3.422 0.  0         3000000    3000000
#             x      y     z    startTime decayTime stopTime
```

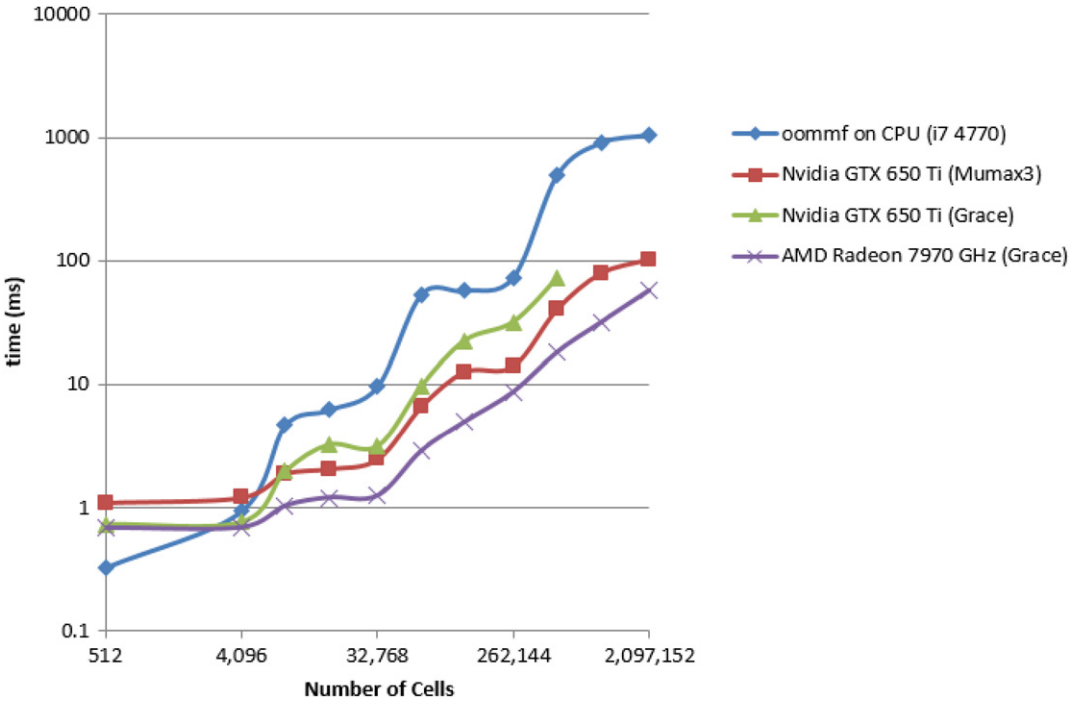

Fig. 6. Time needed to carry out one timestep for different 3D problem sizes. Note the stair-case shape behavior due to slow FFT calculations for non-power-of-two input sizes.

Table 1
Per-step simulation time needed by CPU and GPU solvers for different 3D problem sizes with the Euler algorithm. Numbers are in milliseconds. Mumax version 3.6.2 Windows 64-bit was used in this benchmark. It can be observed that Grace outperformed Mumax for smaller input problem sizes but was about two times slower for large inputs. Due to memory limits, Grace was not able to handle extremely large inputs, indicated by N/A in the last two rows.

Size	CPU i7 4770 (OOMMF)	NVidia GTX 650 Ti (Mumax3)	Speedup	NVidia GTX 650 Ti (Grace)	Speedup	AMD Radeon 7970 (Grace)	Speedup
512 (8^3)	0.325	1.09	0.298	0.733	0.443	0.677	0.480
4096 (16^3)	0.933	1.20	0.778	0.764	1.22	0.683	1.37
8000 (20^3)	4.64	1.87	2.48	1.94	2.39	1.02	4.55
15,625 (25^3)	6.18	2.05	3.01	3.21	1.92	1.19	5.19
32,768 (32^3)	9.59	2.46	3.90	3.14	3.05	1.24	7.73
64,000 (40^3)	53.1	6.57	8.08	9.54	5.57	2.87	18.5
125,000 (50^3)	56.9	12.7	4.48	22.3	2.55	4.94	11.5
262,144 (64^3)	73	14.0	5.21	31.2	2.34	8.62	8.47
512,000 (80^3)	496	40.5	12.24	71.1	6.98	18.1	27.4
100,000 (100^3)	907	79.2	11.45	N/A	N/A	31.4	28.9
2,097,152 (128^3)	1055	103	10.24	N/A	N/A	56.9	18.5

According to Figs. 4 and 5 the average magnetization results and the magnetization distribution from Grace are in good agreement with those from OOMMF [11], the software widely used for a decade. Thus, the result is reliable.

4. Impact

This simulator solves the magnetization dynamics problem from nanometer to micrometer scales, so it has wide appli-

cations in magnetic read/write head design, magnetic recording media modeling, and spin-torque device development. The high simulation speed and low hardware requirements allow large-scale simulations to be carried out within a short period of time and limited budget. Since its public release it has attracted a large number of visits to its download page (https://sites.google.com/site/gracegpu/). According to Google Analytics, an online statistics tool, over 300 users visit the download page every week.

Since the main purpose of this work is to implement GPU acceleration in micromagnetic simulations, a benchmark for the performance of the simulator is demonstrated. A cubic magnetic sample with an exchange constant of $A = 1 \times 10^{-11}$ J/m, a saturation magnetization of $M_s = 1000$ kA/m and an anisotropy field of $H_{anis} = 100$ kA/m was studied. The sample was divided into grids of $N \times N \times N$ and reached its relaxation state by applying the LLG equation to each cell. The testing hardware was an AMD Radeon HD 7970 GHz Edition GPU with an Intel Xeon E5410 CPU. The GPU in use was among the fastest available on the consumer market but still cost less than $500. For comparison, the benchmark data on OOMMF running on an i7-4770 CPU is also shown.

As shown in Table 1 and Fig. 6, large speedup factors were achieved, up to two orders of magnitude for large problem sizes. This suggests Grace has a comparable performance to Mumax [6]. Large problem sizes are frequently seen in simulations of complex magnetic nanostructures (magnetic read/write heads, spin oscillators, etc.), so a simulation with a high accuracy may take days on a CPU, but it can be shortened to hours with the aid of GPU acceleration as described in this paper.

There are other GPU programming platforms available, such as NVidia's Compute Unified Device Architecture (CUDA) and Open Computing Language (OpenCL) developed by Apple Inc. Compared to CUDA, which requires the program to run on a NVidia's GPU, C++ AMP is fully compatible with different hardware platforms, so a program written in C++ AMP can migrate to a different GPU without any modification. OpenCL is also platform independent, but C++ AMP is preferred because it features a simplified Application Programming Interface (API) to make the programming on GPUs easier for developers [17,18].

The computing power of a GPU is considerable greater than 1 Trillion floating point operations per second or TFLOPS for a high-end product), but it is much slower at transferring data between a CPU and a GPU (about 10 GB/s), which is the bottleneck in high-performance GPU computing. To overcome this critical speed bottleneck and maximize the simulation speed of Grace, all the simulation work are done on the GPU where the data I/O is much faster (greater than 200 GB/s) after the initial condition is set up, except for writing data to an output file. The memory latency is hidden by implementing parallel codes on GPUs, including the calculation of the demagnetization, exchange, anisotropy and external fields.

5. Conclusions

To the best of the author's knowledge, Grace is the first hardware-independent implementation of a micromagnetic simulator with publicly accessible code. A speedup factor of up to two orders of magnitude is achieved in large simulations. Thanks to its hardware compatibility, users can now run smaller scale problems on their laptops with integrated graphics to get preliminary results and run large-scale problems on workstations with a professional graphics card to get results with high accuracy. More features will be added to Grace in the future, including the use of non-regular geometry and adaptive timesteps.

Acknowledgment

This work was supported by Graceland University professional development program.

References

[1] Donahue Michael Joseph, Donald Gene Porter. OOMMF User's guide. US Department of Commerce, Technology Administration National Institute of Standards and Technology; 1999.
[2] Scholz Werner, et al. Scalable parallel micromagnetic solvers for magnetic nanostructures. Comput. Mater. Sci. 2003;28(2):366–83.
[3] Hayashi Nobuo, Saito Koji, Nakatani Yoshinobu. Calculation of demagnetizing field distribution based on fast Fourier transform of convolution. Jpn J Appl Phys 1996;35(12A):6065–73.
[4] Yuan Samuel W, Neal Bertram H. Fast adaptive algorithms for micromagnetics. IEEE Trans Magn 1992;28(5):2031–6.
[5] Kakay Attila, Westphal Elmar, Hertel Riccardo. Speedup of FEM micromagnetic simulations with graphics processing units. IEEE Trans Magn 2010;46(6):2303–6.
[6] Vansteenkiste Arne, Van de Wiele Ben. MuMax: a new high-performance micromagnetic simulation tool. J Magn Magn Mater 2011;323(21):2585–91.
[7] Chang R, et al. FastMag: Fast micromagnetic solver for complex magnetic structures. J Appl Phys 2011;109(7):07D358.
[8] Li Shaojing, Boris Livshitz, Vitaliy Lomakin. Graphics processing unit accelerated micromagnetic solver. IEEE Trans Magn 2010;46(6):2373–5.
[9] Lopez-Diaz L, et al. Micromagnetic simulations using graphics processing units. J Phys D: Appl Phys 2012;45(32):323001.
[10] Donahue Michael, et al. μMAG Standard Problem #3. NIST, Mar 1998. Web. Jan.2015.
[11] McMichael RD, et al. Switching dynamics and critical behavior of standard problem No. 4. J Appl Phys 2001;89(11):7603–5.
[12] Zhu Ru, Speedup of Micromagnetic Simulations with C++ AMP On Graphics Processing Units. arXiv preprint arXiv:1406.7459, 2014.
[13] Nakatani Yoshinobu, Yasutaro Uesaka, Nobuo Hayashi. Direct solution of the Landau-Lifshitz-Gilbert equation for micromagnetics. Jpn J Appl Phys 1989;28(12R):2485.
[14] Donahue Michael J, Porter DonaldG. Exchange energy formulations for 3D micromagnetics. Phys. B: Condens Matter 2004;343(1):177–83.
[15] Kate Gregory, Miller Ade. C++ AMP: Accelerated massive parallelism with Microsoft® visual C++®. O'Reilly Media, Inc.; 2012.
[16] Daniel Moth, et al. C++ AMP FFT Library. CodePlex, Jan 2013. Web. 23 Jun. 2014.
[17] Steve Deitz, C++ AMP for the CUDA Programmer. MSDN, Apr 2012. Web. Jan. 2015.
[18] Steve Deitz, C++ AMP for the OpenCL Programmer. MSDN, Apr 2012. Web. Jan. 2015.